

C# Avançado

WPF Parte 1: Criação de Interfaces Gráficas




Softblue
cursos online

Tópicos Abordados



- Windows Presentation Foundation
- Classes *Application* e *Window*
- Linguagem XAML
 - Sintaxe
 - O papel do XAML em aplicações WPF
- Tratamento de eventos
- Controles do WPF
- Gerenciadores de layouts
 - *StackPanel*, *WrapPanel*, *Canvas*, *DockPanel* e *Grid*

Tópicos Abordados



- Outros controles
 - Menu, barra de ferramentas, barra de status, caixas de diálogo e caixa de mensagem
- Resources
 - Tipos
 - Como referenciar

Windows Presentation Foundation



- Antes do .NET 3.0, a API *Windows Forms* era utilizada na criação de interfaces gráficas
 - Era necessário utilizar outras APIs para renderização 2D/3D, vídeo e documentos
- A partir do .NET 3.0, surgiu a API **Windows Presentation Foundation**
 - Chamada de **WPF**
 - Objetivos
 - Unificação de APIs
 - Separação da lógica da aplicação da parte visual
- O objetivo deste módulo é abordar alguns aspectos importantes do WPF

Assemblies Necessários



- O uso de WPF requer a referência a quatro assemblies
 - *PresentationCore.dll*
 - *PresentationFramework.dll*
 - *System.Xaml.dll*
 - *WindowsBase.dll*
- O Visual Studio já cria as referências a estes assemblies no momento da criação do projeto do tipo WPF

As Classes *Application* e *Window*



- São as primeiras classes com as quais você terá contato ao criar suas aplicações gráficas
 - *Application*
 - Representa a aplicação
 - Existe apenas uma instância dessa classe durante a execução
 - *Window*
 - Representa uma janela da aplicação

As Classes *Application* e *Window*



```
class Program : Application
{
    [STAThread]
    static void Main(string[] args)
    {
        Program app = new Program();
        app.Startup += AppStartup;
        app.Exit += AppExit;
        app.Run();
    }

    static void AppStartup(object sender, StartupEventArgs args) { }
    static void AppExit(object sender, ExitEventArgs args) { }
}
```

```
class AppWindow : Window
{
    public AppWindow(string title, int width, int height)
    {
        this.Title = title;
        this.Width = width;
        this.Height = height;
    }
}
```

A Linguagem XAML

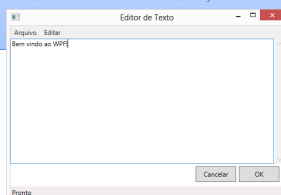


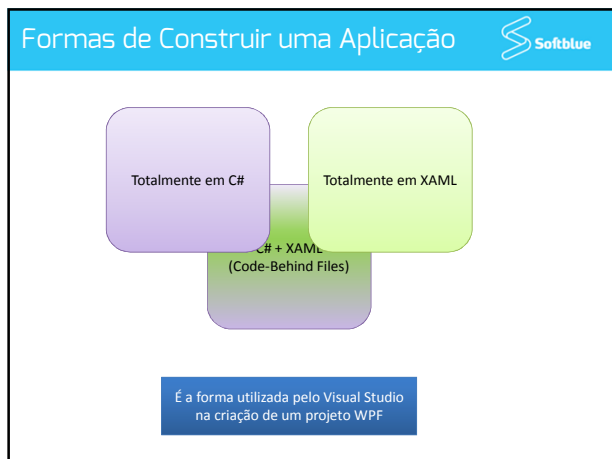
- WPF introduziu a linguagem XAML
 - Extensible Application Markup Language
- É uma linguagem baseada no formato XML
- Utilizada para construir a parte visual da aplicação

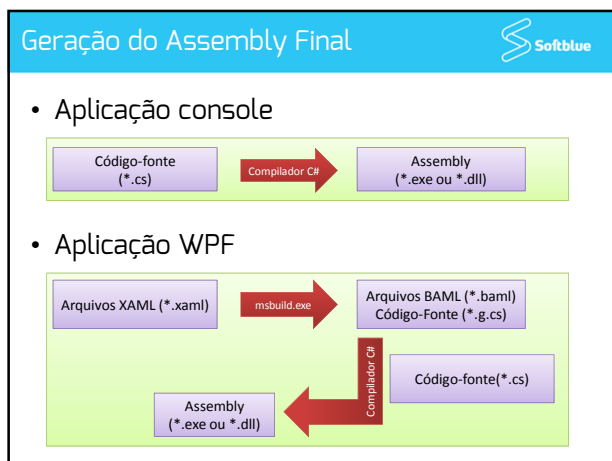
Exemplo da Linguagem XAML

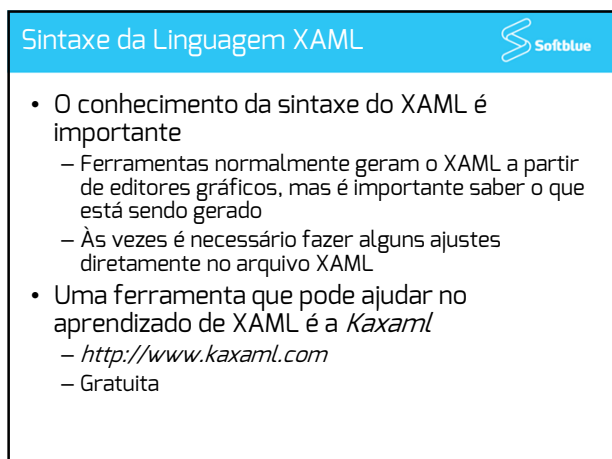


```
<Window x:Class="SoftApplication.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Editor de Texto" Height="358" Width="525">
    <Grid Margin="0,0,0,0">
        <Menu HorizontalAlignment="Left" VerticalAlignment="Top" Width="507" Height="24">
            <MenuItem Header="Arquivo" Height="24"/>
            <MenuItem Header="Editar"/>
        </Menu>
        <Button Content="OK" HorizontalAlignment="Left" Margin="423,261,0,0" VerticalAlignment="Top" Width="75"
            Height="29"/>
        <Button Content="Cancelar" HorizontalAlignment="Left" Margin="343,261,0,0" VerticalAlignment="Top" Width="75"
            Height="29"/>
        <StatusBar HorizontalAlignment="Left" Height="21" Margin="0,298,0,0" VerticalAlignment="Top" Width="507">
            <TextBlock TextWrapping="Wrap" Width="507" Height="21"><Run Language="pt-br" Text="Pronto"/></TextBlock>
        </StatusBar>
        <ScrollViewer HorizontalAlignment="Left" Height="232" Margin="0,24,0,0" VerticalAlignment="Top" Width="507">
            <TextBlock Height="213" TextWrapping="Wrap" Text="Sem vindo ao WPF!" Width="488" HorizontalAlignment="Left"
                VerticalAlignment="Top"/>
        </ScrollViewer>
    </Grid>
</Window>
```








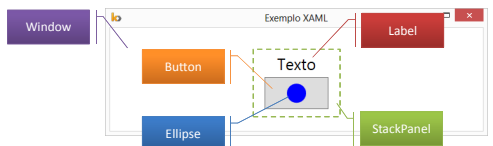


Sintaxe da Linguagem XAML




```
<Window
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  Title="Exemplo XAML" Width="600" Height="200">

  <StackPanel VerticalAlignment="Center" HorizontalAlignment="Center"
    Orientation="Vertical">
    <Label Content="Texto" FontSize="25" HorizontalAlignment="Center" />
    <Button Width="100" Height="50">
      <Ellipse Width="30" Height="30" Fill="Blue" />
    </Button>
  </StackPanel>
</Window>
```



Unidades de Medida



- Usar pixels diretamente pode afetar o resultado visual da aplicação dependendo das características da tela
- Todos os tamanhos são especificados em *device-independent unit*
 - Adequação dos elementos a diferentes resoluções e densidades de telas

Tratamento de Eventos



- A linguagem XAML é capaz de referenciar métodos que são chamados quando da ocorrência de eventos

```
<Window x:Class="MainWindow"
  <Button x:Name="btnApp" Content="Botão" HorizontalAlignment="Center"
    VerticalAlignment="Center" Width="75" Height="37" Click="btnApp_Click"/>
</Window>
```



```
public partial class MainWindow : Window
{
  private void btnApp_Click(
    object sender, RoutedEventArgs e)
  {
    MessageBox.Show("Botão clicado!");
  }
}
```

Controles do WPF



- São elementos usados na montagem da interface gráfica
- Exemplos

Input	Janela	Layout
Button	Menu	Grid
ComboBox	ToolBar	Canvas
TextBox	StatusBar	StackPanel
Label	ProgressBar	Viewbox
CheckBox	ToolTip	DockPanel
RadioButton		Border

- A documentação do .NET fornece detalhes sobre a lista completa de controles

Gerenciamento de Layout com Painéis



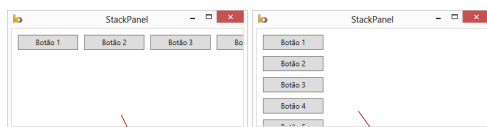
- Dificilmente uma interface gráfica é composta apenas por um controle
 - Normalmente, uma série de controles são utilizados na interface
 - Estes controles são organizados na interface gráfica
- A interface gráfica pode ser organizada através do uso de painéis

StackPanel



- Os controles são dispostos na horizontal ou na vertical

```
<Window>
<StackPanel Orientation="Horizontal">
  <Button Content="Botão 1" Width="100" Height="25" Margin="0,10,10,0" />
  <Button Content="Botão 2" Width="100" Height="25" Margin="0,10,10,0" />
  <Button Content="Botão 3" Width="100" Height="25" Margin="0,10,10,0" />
  <Button Content="Botão 4" Width="100" Height="25" Margin="0,10,10,0" />
  <Button Content="Botão 5" Width="100" Height="25" Margin="0,10,10,0" />
</StackPanel>
</Window>
```



Orientação horizontal

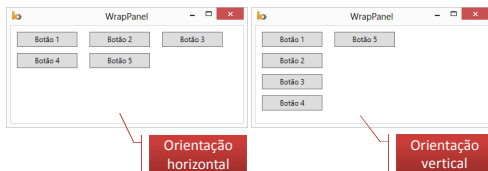
Orientação vertical

WrapPanel



- Funciona como o *StackPanel*, mas arruma os controles se eles não couberem na tela

```
<Window>
  <WrapPanel Orientation="Horizontal">
    <Button Content="Botão 1" Width="100" Height="25" Margin="10,10,10,0" />
    <Button Content="Botão 2" Width="100" Height="25" Margin="10,10,10,0" />
    <Button Content="Botão 3" Width="100" Height="25" Margin="10,10,10,0" />
    <Button Content="Botão 4" Width="100" Height="25" Margin="10,10,10,0" />
    <Button Content="Botão 5" Width="100" Height="25" Margin="10,10,10,0" />
  </WrapPanel>
</Window>
```

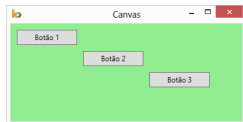


Canvas



- Permite posicionar os controles de forma absoluta na tela

```
<Window>
  <Canvas Width="400" Height="200" Background="LightGreen">
    <Button Content="Botão 1" Canvas.Left="10" Canvas.Top="10" />
    <Button Content="Botão 2" Canvas.Left="120" Canvas.Top="45" />
    <Button Content="Botão 3" Canvas.Left="230" Canvas.Top="80" />
  </Canvas>
</Window>
```



Os controles só aparecem se estiverem na área visível da janela

DockPanel



- Pode fixar os controles nas laterais ou nas partes inferior ou superior

```
<Window>
  <DockPanel Width="400" Height="200">
    <Button Content="Botão 1" DockPanel.Dock="Left" />
    <Button Content="Botão 2" DockPanel.Dock="Right" />
    <Button Content="Botão 3" DockPanel.Dock="Top" />
    <Button Content="Botão 4" DockPanel.Dock="Bottom" />
  </DockPanel>
</Window>
```

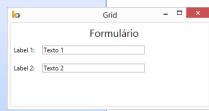


- Permite a disposição dos controles em células, semelhante a uma tabela

```
<Window>
  <Grid Width="400" Height="200" ShowGridLines="False">
    <Grid.ColumnDefinitions>
      <ColumnDefinition Width="60" />
      <ColumnDefinition Width="*" />
    </Grid.ColumnDefinitions>

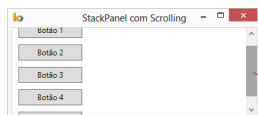
    <Grid.RowDefinitions>
      <RowDefinition Height="35" />
      <RowDefinition Height="35" />
      <RowDefinition Height="35" />
    </Grid.RowDefinitions>

    <Label Content="Formulário" Grid.ColumnSpan="2" FontSize="20" />
    <Label Content="Label 1:" Grid.Row="1" Grid.Column="0" />
    <TextBox Text="Texto 1" Grid.Row="1" Grid.Column="1" />
    <Label Content="Label 2:" Grid.Row="2" Grid.Column="0" />
    <TextBox Text="Texto 2" Grid.Row="2" Grid.Column="1" />
  </Grid>
</Window>
```



- Um *ScrollViewer* pode ser utilizado quando é necessário que um painel tenha scrolling

```
<Window>
  <ScrollViewer>
    <StackPanel>
      <Button Content="Botão 1" Width="100" Height="25" />
      <Button Content="Botão 2" Width="100" Height="25" />
      <Button Content="Botão 3" Width="100" Height="25" />
      <Button Content="Botão 4" Width="100" Height="25" />
      <Button Content="Botão 5" Width="100" Height="25" />
    </StackPanel>
  </ScrollViewer>
</Window>
```

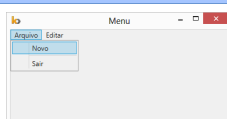


Barra de rolagem

- Menus de opções podem ser criados
- Normalmente ficam na parte superior

```
<Window>
  <DockPanel>
    <Menu DockPanel.Dock="Top" HorizontalAlignment="Left">
      <MenuItem Header="_Arquivo">
        <MenuItem Header="_Novo" />
        <Separator />
        <MenuItem Header="_Sair" />
      </MenuItem>
      <MenuItem Header="_Editar">
        <MenuItem />
      </MenuItem>
    </Menu>
  </DockPanel>
</Window>
```

O "_" define o caractere para atalho via teclado

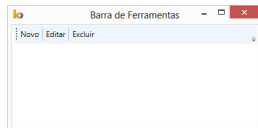


Barra de Ferramentas



- É composta por botões que realizam determinadas ações

```
<Window>
  <DockPanel>
    <ToolBar DockPanel.Dock="Top" Height="30" VerticalAlignment="Top">
      <Button Content="Novo" />
      <Separator/>
      <Button Content="Editar" />
      <Separator/>
      <Button Content="Excluir" />
    </ToolBar>
  </DockPanel>
</Window>
```

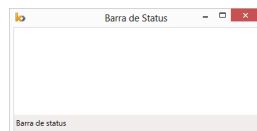


Barra de Status



- É uma barra informativa exibida na parte inferior da janela

```
<Window>
  <DockPanel>
    <StatusBar DockPanel.Dock="Bottom" Height="25" VerticalAlignment="Bottom">
      <TextBlock Text="Barra de status" />
    </StatusBar>
  </DockPanel>
</Window>
```

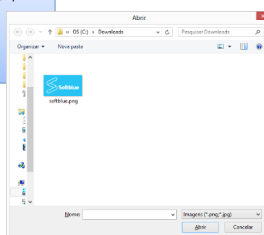


Caixas de Diálogo: Open e Save




- As classes *OpenFileDialog* e *SaveFileDialog* podem ser utilizadas quando é necessário abrir ou salvar um arquivo, respectivamente

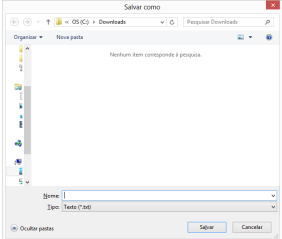
```
OpenFileDialog dialog = new OpenFileDialog();
dialog.InitialDirectory = "C:\\Downloads";
dialog.Filter = "Imagens|*.png;*.jpg";
if (dialog.ShowDialog(this) == true)
{
  String file = dialog.FileName;
}
```




Caixas de Diálogo: Open e Save



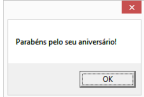
```
SaveFileDialog dialog = new SaveFileDialog();
dialog.InitialDirectory = "C:\\Downloads";
dialog.Filter = "Texto|*.txt";
if (dialog.ShowDialog(this) == true)
{
    String file = dialog.FileName;
}
```



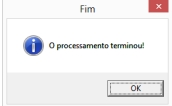
Caixa de Mensagem



- A classe *MessageBox* pode ser usada quando é necessário exibir uma mensagem ao usuário ou fazer uma pergunta




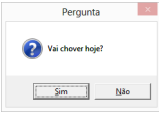
```
MessageBox.Show(this, "Parabéns pelo seu aniversário!");
```



```
MessageBox.Show(this, "O processamento terminou!", "Fim",
    MessageBoxButton.OK, MessageBoxImage.Information);
```

Caixa de Mensagem





```
MessageBoxResult result = MessageBox.Show(this, "Vai chover hoje?",
    "Pergunta", MessageBoxButton.YesNo, MessageBoxImage.Question);
if (result == MessageBoxResult.Yes)
{
    //...
}
else
{
    //...
}
```

Usando Resources



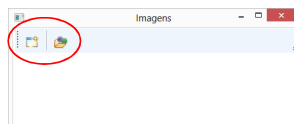
- Resources são arquivos referenciados pela aplicação, utilizados durante a execução
- Exemplos
 - Imagens
 - Sons
 - Vídeos
 - Documentos
 - etc.

Tipos de Resources



- Externos ao assembly
 - Os arquivos ficam no sistema de arquivos, dentro do diretório da aplicação
- Internos ao assembly
 - Os arquivos são incorporados ao assembly da aplicação

```
<Button>  
<Image Source="Images/new.png" />  
</Button>  
<Button>  
<Image Source="Images/open.png" />  
</Button>
```



Resources Via Programação



- Resources podem ser referenciados via programação

Externos ao assembly

```
string dir = Environment.CurrentDirectory;  
imgNew.Source = new BitmapImage(new Uri(dir + "\\Images\\new.png"));  
imgOpen.Source = new BitmapImage(new Uri(dir + "\\Images\\open.png"));
```

Diretório da aplicação

Internos ao assembly

```
imgNew.Source = new BitmapImage(new Uri(@"\\Images\\new.png",  
UriKind.Relative));  
imgOpen.Source = new BitmapImage(new Uri(@"\\Images\\open.png",  
UriKind.Relative));
```

Diretório relativo